METHOD OF COMBINATORIAL MULTIMODAL OPTIMISATION

The present invention relates to methods of combinatorial multimodal optimisation, and in particular to the use of genetic algorithms for solving combinatorial multimodal optimisation problems. The invention is expected to find particular although not exclusive application in the fields of load balancing (e.g processor and network load balancing), scheduling optimization (including production scheduling, resource assignments, timetable scheduling) and resource planning.

An optimisation problem is always a simplication of a real world problem, an engineer needs to suggest various possible optimal solution alternatives to the decision maker in order to understand this problem better.

Many problems in the scheduling of manufacturing systems and artificial intelligence can be regarded as combinatorial multimodal optimisation problems, which require the determination of multiple integer solution vectors that maximise (or minimise) a given objective function with regard to some definite set of constraints. Problems of this class, which are characterised by their discreteness, high dimension and noise, are quite different from multimodal continuous function optimisation problems. The combinatorial multimodal optimisation problem is arguably one of the most deceptive problems for GA applications.

So-called genetic algorithms (GA) are well-known stochastic optimisation search methods which are based on a global search procedure. But the use simple genetic operators (selection, multi-point crossover and mutation) to

solve a typical combinatorial multimodal optimisation problem, provides only a single (global or local) optimal solution for each search procedure [He Liwen and Mort. Neil: Genetic Algorithm for Scheduling Optimization on a Multi-Product Batch Processing Machine. *4th IFAC Workshop on Intelligent Manufacturing Systems,* Seoul, Korea, PP165-170, 1997]. The reason is that, as there is no selective pressure among any of the peaks for the finite population in the fitness landscape, the population converges to one alternative or another randomly. This problem is known as genetic drift -- stochastic errors in sampling caused by small population size. In other words, this difficulty comes from the fundamental theorem of genetic algorithms which tells us that exponentially increasing trials will be given to the best schemata observed, thus losing some potentially useful schemata during the evolution.

Therefore, the problem is how to reduce the effect of this genetic drift and keep the population diversity in GA search in order to find different fitness peaks, and lead populations to move from local optima to global optima.

A typical NP-hard Combinatorial Optimisation Problem could be described as follows [Jelasity, M. and Dombi, J: GAS, a Concept on Modelling Species in Genetic Algorithms. *Artificial Intelligence,* Vol. 99, No. 1, pp. 1-19, 1998]:
A set $W = \{w_1, w_2, \ldots, w_n\}$ of n integers and a large integer P is given. We would like to find an $S \subseteq W$ such that the sum of the elements in S is close to, without exceeding, P, perhaps subject to some problem-specific constraint conditions. The number of elements in S is given as m.

To take a simple example, let us assume that we have eleven jobs of various sizes that need to be shared, within a computer system, between five parallel

processors. Each processor is capable of handling a single job of size 9, or any combination of smaller jobs having a total aggregate size of more than 9.

In this example, let us assume that the eleven jobs are defined by the set $W = \{2, 7, 5, 4, 3, 6, 2, 4, 2, 8, 1\}$ of 11 integers; using the above terminology, $n = 11$, $m = 5$, and $P = 9$. The task, then, is defined 5 groups of jobs such that the sum of the values within each group is close to, but does not exceed, 9.

One possible solution is as follows:

Processor 1: {2, 7}
Processor 2: {5, 4}
Processor 3: {3, 6}
Processor 4: {2, 4, 2} and
Processor 5: {8, 1}

With this solution, four of the processors have an aggregate workload of 9, while the remaining processor (processor 4) has an aggregate work value of 8. Since all of these workloads are sufficiently close to but not exceed the maximum value of 9, we say that the *fitness value* of this possible solution is 5. Since that is the same as the number of processors, this solution is optimal.

Other trial solutions may give less good results. In the following groupings,

Processor 1: {2, 7}
Processor 2: {5, 4}
Processor 3: {3, 6, 2}

Processor 4: {4, 2}; and

Processor 5: {8, 1}

only the first, second and fifth processors have a full workload. The third processor has a workload of 11 and is so overextended, while the fourth processor has a workload of 6 and is under-utilised. Since only three of the five groupings meet the problem criteria, the *fitness value* of this possible solution is 3.

If the number of the global optima is greater than 1 in the solution space, then this problem is regarded as combinatorial multimodal optimisation problem.

Conventional genetic algorithms start with an initial population of individuals (trial solutions). The fitness value of each individual is evaluated according to some function (sometimes called the "objective function"), and there then follows a solution process under which individuals with higher fitness values have a higher probability of surviving into the next generation. Typically, individuals with low selection values are eliminated entirely, and individuals with high selection values are replicated several times, in proportion to their fitness value. Genetic operators such as crossover and mutation are then applied to the population, and the fitness values of the new individuals are re-evaluated. This process continues until the algorithm converges.

As mentioned previously, such an approach is limited because of the problems of genetic drift, and the fact that only a single optimal solution can be obtained. Indeed, the very nature of the procedure tends to ensure that sub-optimal individuals (in terms of fitness value) ultimately become reduced and

eliminated from the population, so ensuring that as the algorithm proceeds only a smaller and smaller area of the solution space is explored. That may be useful for finding local optima, but at the expense of finding global optima.

Several approached have been proposed to deal with these problems, such as niching methods, fitness sharing, [Sareni, B and Krahenbuhl, L: Fitness Sharing and Niching Methods Revisited. *IEEE Transactions on Evolutionary Computation*. Vol. 2, No. 3, 1998] and parallel genetic algorithms [Chipperfield A.J and Fleming, P.J: Parallel Genetic Algorithms: A Survey. Research Report No. 518, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994]. Other techniques include those proposed by Sirinivas, M. and Patnaik. L.M: Adaptive Probabilities of Crossover and Mutation in Genetic Algirithms. *IEEE Transaction on Systems, Man and Cybernetics*. Vol. 24, No. 4, 1994; Schneider. G., Schuchhardt. J. and Wrede. P: Evolutionary Optimization in Multimodal Search Space. *Biological Cybernetics*. Vol. 74, No. 3, pp.203-207, 1996; Yao. X and Liu. Y: Fast evolution strategies. *Control and Cybernetics*. Vol.26, No. 3, pp.467-496, 1997; and Jelasity, M. and Dombi, J: GAS, a Concept on Modelling Species in Genetic Algorithms. *Artificial Intelligence*, Vol. 99, No. 1, pp. 1-19, 1998.

Both niche techniques and adaptive GA require a previous understanding of the number and the distribution of fitness peaks, and need careful setting of various parameters. Parallel GA has proven very reliable and efficient in exploring multimodal fitness peaks, and it also needs fewer assumptions than niche methods. The main limitation of implementing Parallel GA is the sub-population size which limits the total number of global optima found in the search.

According to the present invention there is provided a method of combinatorial multimodal optimisation for finding multiple optimal ways of dividing a set W of n values into m groups, such that each of the groups satisfies a respective constraint condition, the method comprising:

(a) defining an initial population of individuals, each representative of a trial solution;

(b) calculating for each individual a fitness vector indicative of whether the constraint condition for each group has been satisfied;

(c) selecting a plurality of individuals for the next generation in dependence upon their respective fitness vectors;

(d) creating a new population including the selected individuals; and

(e) repeating steps (b) to (d) until the population stabilizes, the individuals of the stable population representing multiple optional ways of dividing the set W.

Experimental results show that both convergence speed and quality are greatly improved in comparison with other GA techniques. They also suggest that this novel method has the capability of generating sufficient genetic innovations, preserving promising genetic diversity, guiding population escape from local optima and exploring multiple global optima effectively and efficiently in this difficult combinatorial multimodal optimization problem.

The novel method works through deeply preserving the population diversity based on a phenotypic analysis, rather than simply a genotypic analysis. This reinforces the power of the method to process many building blocks in the

search space simultaneously. This power is stronger than that of the implicit parallelism found in simple genetic algorithms, and thus enhances the exploration of GA space for multiple optima.

In this novel method a hybrid drift model, through the complementary interaction between genetic drift and neutral drift, can help a population escape away from local optima. When the relative fitness vector of one individual is non-dominated by that of any other one in the entire population, 'neutral drift' is in effect. Any non-dominated individuals are regarded as having equal-(relative)-fitness values and kept for the next generation, thus facilitating the exploration of the different adaptive peaks. On the other hand, when one individual is dominated by another, 'genetic drift' effectively lets this less fit individual die out. This model has been shown to be effective in leading populations to escape from local optima to global optima in this typical combinatorial multimodal optimisation problem.

The invention may be carried into practice in a number of ways and one specific example will now be described, merely by way of example.

Briefly stated, the preferred method of the present invention makes use of a genetic algorithm having the following steps:

1. An initial population of individuals is randomly generated, each individual being represented by a string of integers which defines a trial solution;
2. A fitness value and fitness vector is calculated for each individual, according to an objective function;

3. On the basis of the fitness values and the fitness vectors, suitable individuals are selected to form a sub-population whose size is a portion of the overall population;
4. A roulette wheel selection method is then used to create offspring, from the selected individuals, to form a new population;
5. Highly uniform crossover and mutation is applied to the individuals of the new population; and
6. Steps 2 to 5 are repeated until the algorithm converges.

Each of these individual steps will be described in more detail below.

First, we will consider representation and initialisation of the population. A major difficulty in applying GA in a scheduling problem is finding an appropriate representation for the population. Here, an integer string representation of the individual is selected according to the following conditions:

a. *Completeness*: the string representation should contain all possible individuals in the search population.
b. *Uniqueness*: the string maps to the individual on a one to one basis.

A second desideratum is to generate the initial population randomly. For example, a population is composed of N_{chrom} individuals each of which is an integer string with n genes. The initial population is produced by creating a $(N_{chrom} \times n)$ matrix with the component randomly selected from the set {1, …, m}.

In contrast with the prior art, the fitness of each individual is determined not only on the basis of the fitness value but on that individual's *fitness vector*.

The fitness vector is a binary vector of length m, the individual bits of which indicate, for that individual, whether the groupings satisfy or do not satisfy the problem constraints for each of the m groupings.

To go back to the example previously discussed, in which 11 parcels of work have to be divided between five processors, such that no processor takes more than 9 work units, the first solution given satisfied the constraints for all of the five processors. Its fitness value was therefore 5, and its fitness vector FV = [1 1 1 1 1]. The second, less-good solution, satisfied the constraints only for the first, second and fifth processors, and hence had a fitness value of 3. The fitness vector in that case becomes [1 1 0 0 1].

It will be understood of course that, with this approach, the fitness value is simply equivalent to the number of positive bits within the fitness vector.

We now introduce some additional terminology, to be used in the remainder of the description. GA applications sometimes refer to the individuals within the evolving population as *genotypes*. Following that analogy, we shall refer to analysis using the fitness vector as phenotype or phenotypic analysis. We define the groups of combinations making up the solutions, for each individual, as a *phenotype schema.*

To clarify this further, it may be useful to consider a further example which comes from a practical shoe batch processing machine (BPM) optimization problem. This problem is described in Fanti, M.P, Maione, B, Piscitelli, G and Turchiano, B: Heuristic scheduling of jobs on a multi-product batch processing

machine, *International Journal of Production Research,* **34,** pp. 2163-2186, 1997. In this problem, n is 21, m is 7, P is 811, and the vector W is defined by:

[12 12 170 176 216 231 266 310 380 497 714 144 128 153 660 688 50 6 454 282 114]

Thus, the problem comprises dividing up the elements of W into seven groups, such that the sum of the elements in each group is close to but does not exceed 811. We will now consider how, in the preferred embodiment, any fitness value and fitness vector is calculated for an individual or trial solution which is defined by an integer string b, of length 21, as follows:

[4 5 3 7 7 4 4 1 5 1 2 6 7 5 6 2 3 6 3 7 4]

First, the twenty one integers are mapped to a trial phenotypic solution $\mathbf{M}_1$ matrix. $\mathbf{M}_1$ has seven columns, one for each of the seven possible groups, as follows:

Group	1	2	3	4	5	6	7
M_1	310	714	170	12	12	144	176
	497	688	50	231	380	660	216
	0	0	454	266	153	6	128
	0	0	0	114	0	0	282
Sum	***807***	1402	674	623	545	***810***	***802***
Phenotype schema	s_1					s_2	s_3

As may be evident, $\mathbf{M}_1$ is filled with the values taken from W, those values going into the columns defined by the integers of the string b. Thus, the first

value (12) within W goes into column 4, the second value (12) goes into column 5, the third value (170) goes into column 3, and so on. If a column already has a number in it, the next number is simply placed below.

As shown above, the columns are then summed, and each sum is tested against the problem constraint conditions to see whether it complies or not. The problem constraint conditions will of course vary from problem to problem, but in this particular example the conditions are that the phenotype M_1 for the string b should satisfy the two following conditions:

$$\text{sum}(s_j) \leq p = 811$$

$$\text{and} \sum_{j=1}^{Nc} (p - \text{sum}(s_j)) \leq p \times m - \text{sum}(W) = 811 \times 7 - 5663 = 14$$

$$\text{for } j=1,2,3$$

where N_c is the number of phenotype schemata (here 3) included in the phenotype M_1, and s_j is the j-th phenotypic schemata.

In the table above, s_1, s_2 and s_3 are the phenotype schemata which satisfy the constraint conditions. Since there are three of those, the fitness value of the string b is 3. The positions of the schemata mean that the fitness vector for b is [1 0 0 0 0 1 1].

More formally:

- Assume that there are a total q of different phenotypic schema $s_1, s_2, \ldots, s_q$ searched in a population, the *phenotype schemata vector* $M = [s_1, s_2, \ldots, s_q]$

is built up as a database of all searched phenotypic schemata in the entire population.

- The phenotype of individual *j* can be mapped into its relative fitness binary vector $fv(j)$ as a binary string in relation to the schemata vector M according to the following principle:

- $fv(j) = [b_1\ b_2\ \ldots\ b_q]$; and b_i ($i = 1, \ldots, q$) is a binary bit ,

$$b_i = \begin{cases} 1 & \text{if the phenotype of individual j includes the schema } s_i ; \\ 0 & \text{if the phenotype of individual j does not include the schema } s_i \end{cases}$$

To take a final example, the table below shows the phenotypic schema, fitness value and corresponding fitness vector for four individuals j_1, j_2, j_3, j_4, and 5 searched phenotypic schemata, namely **a, b, c, d, e**:

Individual	Phenotypic schema	Fitness value	Fitness vector
j_1	{ a , d }	2	(1, 0, 0, 1, 0)
j_2	{ b , c , d , e }	4	(0, 1, 1, 1, 1)
j_3	{ a , c , d , e }	4	(1, 0, 1, 1, 1)
j_4	{ a , b , c , d , e }	5	(1, 1, 1, 1, 1)

Once the fitness value and fitness vector have been calculated for each of the individuals within the population, a selection process takes place to choose which of those individuals are to be retained and which discarded. The selection process is carried out in two stages:

1. First, a reserved sub-population of the best individuals is kept. Individuals are chosen for this sub-population on the basis of their fitness

vectors (and possibly their fitness values as well). One copy only of each individual is retained, thereby ensuring that none of the best individuals within the previous generation can be lost. Typically, the size of this sub-population is limited to around 20% of the total population size.

2. Next, the remaining 80% of the spaces are filled, using a so-called roulette wheel section method on all of the existing individuals. Using this method, offspring for the next generation are created on the basis of the fitness values of the individuals in the current generation. The higher the fitness value of an individual, the more offspring (copies) of that individual will be retained in the next generation. Preferably, the number of offspring is substantially proportional to the fitness value.

In order to understand how the reserved 20% sub-population is filled, it is necessary to introduce the concept of Pareto Optimality: [Goldberg, D.E: *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, MA: Addison Wesley, 1989]. Using this concept, it may be stated that a vector **x** is partially less than ***y***, symbolically $x <_p y$, when the following conditions hold:

$$x <_p y \Leftrightarrow (\forall_i)(x_i \quad y_i) \wedge (\exists i)(x_i < y_i)$$

Under these circumstances we say that vector **x** *dominates* vector **y**. If a vector is not dominated by any other, we define it as *nondominated.*

Turning back to the previous table, giving fitness vectors for the individuals j_1, j_2, j_3, j_4,, we can say that individual j_1 is dominated by both individual j_3 and j_4,

but individual j_2 and j_3 are not dominated with each other. Individual i_4 is not dominated by any other individuals in the population: it is a *nondominated individual* in this population.

Different non-dominated individuals should be kept in the next generation after selection. Actually, most of these non-dominated individuals contain few schemata; they are inferior individuals with relatively smaller absolute fitness values, and might not continue to evolve. So these non-dominated individuals are ranked according to their absolute fitness value: non-dominated individuals with greatest absolute fitness value *Ma* so far are ranked as 'first class' individuals, and others with absolute fitness value (*Ma-k*) (for positive integer k=1, 2,...; and k≤Ma) are ranked as (*k+1*) class individuals. The value of *k* is determined by the initial population size and the number of non-dominated individuals; the greater the value of *k*, the more genetic diversity will be preserved, and the more computation complexity will be required. In this case study, k is chosen as 1, which means that only first-class non-dominated individuals searched so far are guaranteed to be preserved in the next generation in order to maintain suitable genetic diversity and accelerate convergence speed.

The number of non-dominated individuals is preferably fixed to a portion of population size, otherwise no space is available for new entries in the population. The size of non-dominated individuals is preferably set as 20% of population size.

Once the reserved sub-population has been determined, the remaining 80% is filled, as mentioned above, using roulette wheel selection: Goldberg, D.E.:

Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, MA: Addison Wesley, 1989. In this selection, bias optimum is used as the reproduction operator. The number of expected individuals is given by:

$$\text{integer}\left(\frac{Fitness Value of Individual}{\sum Fitness Value of Individuals} \times \text{Population size}\right)$$

Once the new population has been chosen, crossover and mutation is applied to it.

The basic forms of crossover are single-point, double-point, multi-point and uniform crossover. Single-point crossover creates two offspring from two parents. The parents are randomly selected from population, the crossover site, c_x, is selected at random, and two offspring are made by both concatenating the bits that precede c_x in the first parent with those follow c_x in the second parent and performing the obverse operation.

For multi-point crossover, multiple crossover points are chosen at random with no duplicates and sorted in ascending order. Then, the variables between successive crossover points are exchanged between the two parents to produce two new offspring. The section between the first variable and the first crossover point is kept between individuals. The following example illustrates this process.

Consider the following two individuals with 11 binary variables each:

individual 1 0 1 1 1 0 0 1 1 0 1 0
individual 2 1 0 1 0 1 1 0 0 1 0 1

cross points =3 3 6 10

After crossover the new individuals are created:

offspring 1 0 1 1| 0 1 1|1 1 0 1| 1
offspring 2 1 0 1| 1 0 0|0 0 1 0| 0

The disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favouring the convergence to highly fit individuals early in the search, thus making the search more robust.

In uniform crossover, a crossover mask which is a string of binary bits with the size of the chromosome, is generated randomly, the value of each bit in the mask determines for each corresponding bit in a child, which parent it will inherit that bit from. An example is given to illustrate the process.

Parent 1:	2 5 1 4 3 6
Parent 2:	4 6 2 1 5 3
Mask:	0 1 0 0 1 1
Child 1:	2 6 1 4 5 3
Child 2:	4 5 2 1 3 6

Mutation is used as a secondary operator which is randomly applied with small probability. It is a safety policy to be used sparingly with reproduction and crossover against premature convergence to a local optimum.

Two mutation operations are used here. One is "swap" where the two integers at two randomly selected distinct locations of the chromosome (individual) are exchanged. The other is "jump" where the integer in a randomly selected bit of the chromosome jumps to another feasible integer value.

The mutation probability represents the frequency of applying the mutation operator. The mutation rate for jump method is determined by the following condition

$$m_u = \frac{1}{bitlength}$$

In addition to the previously mentioned techniques, further improvements can be achieved by including the following refinements:

- Elitist selection strategy: the best individual is always kept in the next generation (see *Eshelman. L.J: The CHC adaptive search algorithms: How to have safe search when engaging in non-traditional genetic recombination. Foundation of Genetic Algorithms. Gregory J.E. Rawlins (Eds), Morgan Kaufmann Publishers, pp.265-283, 1991*). Also, the stochastic universal sampling method is used to reduce sampling bias (see *Baker. J.E: Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms, pp14-19, 1987*).

- A highly disruptive crossover method (*Eshelman, op cit*) is used to increase the GA capability to exploring a broader search space to find prospective zones which may lead to a new fitness peak.

The method of the preferred embodiment may be implemented, without further inventive input, by a skilled programmer on the basis of the description given above. The algorithm may be conveniently implemented along the lines of the following pseudocode:

```
Initialisation of population
  Generation = 0
  While termination condition not satisfied DO
    Construct phenotypic schemata vector M, fitness vector fv, through
      phenotypic cluster analysis
    Calculate the fitness value of all individuals
    Select suitable non-dominated individuals to form a sub-population whose
      size is a portion of overall population
    Select offspring to form new population using Roulette wheel selection
      method
    Apply highly uniform crossover and mutation to the new population
    Put all the sub-population into the new  population
    Generation = Generation + 1
```